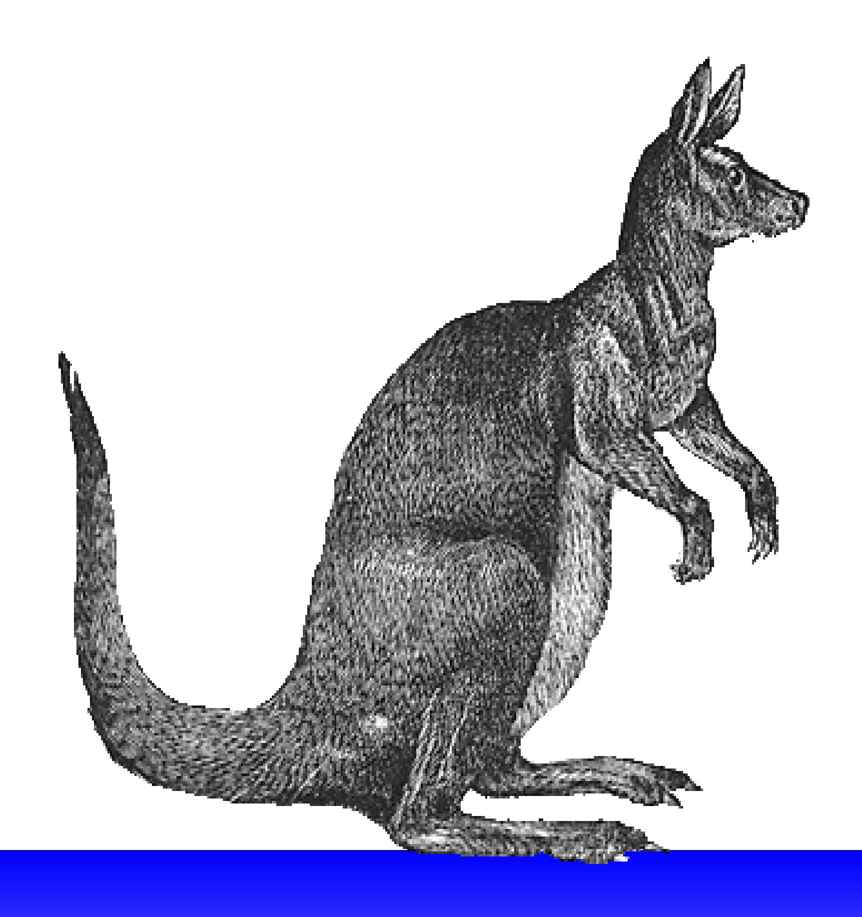


تعلم كتابة قواعد الـudev

أسرع طريقه لتعلم إنشاء الأجهزه الإفتراضيه

قسم الأبحاث التعليميه و العلميه بمجتمع لينكس العربي

2009/2010

المؤلف

م / سيف أبـــاظة

من حق أي شخص الإستفاده من هذا البحث و الإضافه إليه بدون الرجوع إلى المؤلف

بحث تعليمي

جدول المحتويات

بحث تعليمي

. تعلم طريقة إنشاء أجهزة إفتراضيه

• **من تأليف : م / سيف أباظة**

للإتصال بالمؤلف :
مصر : 0127535754
السعوديه : 0595670509
بريد إلكتروني : seif-abaza@hotmail.com
من أهل مجتمع لينوكس العربي : abaza@linuxac.org

هذا البحث هو شرح طريقة إستخدام udev المسؤل عن إنشاء أجهزه ديناميكيه في مجلد الـ /dev

أهم النقاط المُستفاده من البحث

- **التعرف على فلسفة تسميات الأجهزه و تغيرها**
- **طريقة كتابة أسماء جديده للأجهزه و الإحتفاظ بالأسم الجديد**
- **طريقة كتابة القواعد الخاصة بالأجهزه وسنستعرض بعض من هذه القواعد**
- **تشغيل برامج مجرد إرتباط جهاز خارجي بجهاز الكومبيوتر**
- **تغير الصلاحيات و أسم المالك للجهاز**
- **إعادة تسمية الـ Nerwork Interface**

1. المقدمة

كثير من المستخدمين المبتدئين و المحترفين يقعون في فخ أسماء الأجهزه , فالكثير منا يعلم أن ترتيب الأقراص الصلبه (HDD) في نظام تشغيل جنو/لينوكس يأخذ ترتيب أبجدي مثل الـ a , b , c , d مع إسباقها بإختصار مثل hd او sd يعود هذا إلى نوع القرص الصلب (HDD) , لكن تخيل معي أكثر مشكله تُصادف الكثير من المستخدمين وتكون الخسائر بها فادحة و هي , أنه من المعلوم أن القرص الرئيسي هو hda إفتراضاً , فإذا وضعنا قرص أخر خارجي فقد أصبح لدينا الأن hda و hdb , ثما إذا قمنا بوضع قرص أخر فسيكون الناتج لدينا هو hda , hdb , hdc

ونفترض الأن أننا نُريد تهيئة القرص hdc ,و الأسرع و الأفضل هو أن نقوم بالتهيئة من الطرفيه , فإذا سهونا عن أننا نمتلك ثلاث أقراص وليس قرصين فقد نقوم بتهيئة hdb بدلاً من hdc , تكون المشكله إذا كان hdb يحمل بيانات هامه .

من المشاكل الأخرى التي تسبب ضيق لبعض المستخدمين الجدد والذين يُريدون عمل برامج مثلاً للتعامل مع الأجهزه هو فرضاً أن لديك جهازين يعملان بالـ USB , الأول للكميره رقميه Digital Camera و الثاني عباره عن Flash Disk , وعادتاً ما يتم تعريف الإثنين في مجلد dev كتالي

/dev/sdb و /dev/sdc , لكن السؤال الأن أي منهم الكميره و أي منهم الـ Flash Disk ؟

والأفضل أن نقوم بعمل تسميات كتالي حتى نتجنب الحيره في هذا الأمر ,/dev/camera و /dev/flashdisk .

ولكن ما يجب معرفته أن كتابة القواعد ليسة سهله ومختصره مثل المشاكل التي تحدث عند عدم وجود device node لجهاز موجود بالفعل و مُركب عندك .

فالأمر يختلف تماماً , فكل مهمة الـ udev هي إنشاء device node بأسم إفتراضي بإذن من النواه .
و الإحتفاظ بأسم دائم للـ device node له العديد من الفوائد كما ذكرنا .

1- إذاً كيف نكتب القواعد ؟

قبل كل شيء يجب أن نفهم الأساسيات

1.1- التخطيط لعمل أسم دائم للأجهزه

بإمكانك إستخدام الأسماء الإفتراضيه التي يقوم udev بتزويد المستخدم بها , و تكون حل جيد للمستخدم , و الأكثر من هذا أنه لا داعي لكتابة قواعد

فالـ udev يقوم بإنشاء أسماء إفتراضيه للأقراص الصلبه , وتكون موجوده في dev/disk , و لمعرفة تفاصيل عن هذه الأسماء بإمكانك كتابة الأمر التالي في الطرفيه

```
ls -Rl /dev/disk
```

فهذه كل أسماء الأقراص التي تعمل الأن على جهازك , فعلى سبيل المثال نجد أن udev على جهازي قد قام بإنشاء

dev/disk/by-id/ata-Hitachi_HTS541612J9SA00_SB2D02E4K1PKNH

فهذه الأسم الحقيقي للقرص الرئيسي sda , إذا هو عباره عن وصلة Symbolic Link إلى ملف /dev/ بأسم sda , بينما عندما قمنا بوضع الـ Flash Disk , فستكون النتيجه هي كتالي .

/dev/disk/by-id/usb-WD_3200BMV_External_57442D575848583038343535353139-0:0

و هذا أيضاً الأسم الحقيقي للـ Flash Disk و لها Symbolic Link إلى مجلد dev بأسم sdb .
وسنفهم ماذا نقصد بكلمة الأسم الحقيقي ولكن في مرحله متقدمه من هذا البحث .

1.2- الملفات الخاصة بالقواعد و طريقة الصياغة

هناك بعض الأسئلة التي ستقودنا إجابتها إلى كشف الكثير من فوائد الـ udev مثل

- أين تُكتب القواعد ؟

udev يقوم بتنفيذ سلسله من القواعد توجد في /etc/udev/rules.d , وتكون جميعهم بإمتداد .rules

- أين توجد القواعد الإفتراضيه ؟

توجد في المجلد السابق ذكره في السؤال السابق بأسم udev.rules غالباً برقم 50 , ويُستحسن إلقاء نظره على هذا الملف , فهو يحتوي على بعض الأمثله و القواعد التي تعمل , لكن ينبغي الحذر من عدم التعديل على هذا الملف بصوره مُباشره , فله طريقة للتعديل سيتم ذكرها لاحقاً .

- هل هناك ترتيب يجب مُراعته في أرقام ملفات القواعد ؟
الملفات في rules.d مرتبه ترتيب مُرقم وفي بعض الحالات يكون الترتيب على حسب الأهميه , ولكن إذا كنت تُريد أن تكتب قواعدك الشخصيه فمن المُستحسن ترقيمه بأقل رقم أي إذا كان أصغر رقم في هذا المجلد رقم 20 فإجعل ملف قواعدك رقم 19 وهكذا .

- هل هناك علامه للتعليقات ؟
نعم – فتعتبر علامة # تعليق , خلاف هذا فهي قاعدة .

- هل للقاعدة شكل مُعين ينبغي الإلتزام به ؟
نعم – أهم شيء هو عدم كتابة القاعدة في أكثر من سطر , بل يجب كتابتها عى سطر واحد .

- هل من الممكن أن يتطابق الجهاز الواحد مع أكثر من قاعده ؟
نعم – وهذه تعتبر ميزه عمليه في الـ udev , فبإمكانك كتابة قاعدتين للتوافق مع جهاز واحد , حيث أن كل قاعده تُعطي الأسم البديل للجهاز فكل من الإسمين سيتم إنشائهم , حتى إذا كان ملف القاعدة الأولى منفصل عن ملف القاعدة الثانيه , فمن الضروري فهم أنه لم يتوقف الـ udev عن مُعالجة و تنفيذ القواعد مادامت مُتوافقه مع الجهاز , بل سيقوم أيضاً بالبحث و المحاوله لقبول كل القواعد التي يُصادفها أمامه .

1.3- أساسيات كتابة القواعد

القواعد بصوره عامه عباره عن مجموعة مفاتيح تُكون ملف يُسمى ملف rules , وتتكون الـ rules من زوجين من المفاتيح الهامه يُفصل بينهم بفاصله (,) وتكون الصيغه بهذا الشكل
key , value

هذه المفاتيح الأولى أسمها match-key و الثانيه تُسمى Assignment-key .
الـ match-key هي عباره عن مُعرفات ثابته (identfy) وهي من سيُطبق عليها الشروط , فهي تقوم بتحديد أي الأجهزه التي سينطبق عليها القواعد (rules).

المثال التالي يُوضح الطريقة الأساسيه لكتابة rules

KERNEL==”hdb” , NAME=”My-Hard-Disk”

الكلمة KERNEL تُمثل المفتاح Mach-Key , بينا كلمة NAME تُمثل المفتاح Assignment-Key , ولأسماء هذا المفاتيح قواعد وكلمات محددة سنتحدث عنها لاحقاً في هذا البحث .

أهم شيء يجب مُلاحظتة هو أن الـ Match-Key يأتي بعده علامة المساواه مُزدوجة (==) دائماً , بينما الـ Assignment-Key يأتي بعده علامة مُساواه واحده (=) , وما يجب معرفته أيضاً أن الـ udev لا تدعم الـ Line Continuation أي تقبل مسافة واحده فقط بيضاء ولا يجوز إستخدام مسافات مثل الـ Tab إلى في حالات محدوده , أيضاً لا تُكتب الـ rules الواحده على سطرين(أي بالضغط على زر Enter) , فسيعتبرها udev قاعدتين وهذا من المُأكد أنه سيأدي إلى نتائج غير مرغوب بها .

1.4- أساسيات القواعد

يقوم udev بتزويدنا بالعديد من الـ Match Keys الجاهزه للإستخدام لكتابة القواعد التي تطابق الأجهزة بدقة شديدة , بعض المفاتيح الأكثر شيوعا سنستعرضها الأن ، وسوف يتم عرض آخرين في وقت لاحق في هذه الوثيقة للحصول على قائمة كاملة من المفاتيح .

بحث تعليمي

مفتاح Match تستخدم للأجهزه التي تُعرف من خلال النواه	KERNEL
مفتاح Match تستخدم لأجهزة النظم الفرعيه	SUBSYSTEM
مفتاح Match يستخدم لأسم الجهاز الذي يُمنح للجهاز نفسه	DRIVER

نأتي بعد ذالك إلى مفاتيح Assignment-Key حيث أنها تُقدم المزيد من التحكم و المزيد من السيطره على جميع الأجهزه , و الجدير بالذكر أن هناك العديد من أنواع هذه المفاتيح , ولكننا سنستعرض أهمها و التي تُستخدم بكثره .

الأسم الذي ستقوم بإستخدامة للـ Device Node	NAME
إختصار يكون بديل أخر لأسماء الـ Device Node	SYMLINK

الملاحظة التي يجب عليك الإنتباه لها هي , أن udev يقوم بإنشاء device node واحد لجهاز واحد , أي لا يقوم بإنشاء العديد من الـ Device Node لجهاز واحد , وإذا كنت تُريد إنشاء أسماء أخرى للـ Device Node , فستقوم بإستخدام الوظيفة Symlink , ففي الحقيقة كل ما ستفعله هو طلب إختصار للجهاز المطلوب تغير أسمه , أي أن كل الأسماء ستقودك إلى الجهاز المحدد . ولكي تقوم بتعديل القائمة سنحتاج إلى إضافة مُعامل بعد كلمة SYMLINK وهو =+ , كما أنه بإمكانك إلحاق أي عدد من الـ Symlink في ملف الـ rules مع الفصل بينهم بمسافه واحده بيضاء .

```
KERNEL=="hdb" , NAME="My_Spare_Disk"
```

القاعده السابقه تقول : الجهاز الأتي من النواه والذي يتمثل في المفتاح Match و الذي أسمه hdb , سيكون أسمه الجديد My_Spare_Disk و سيظهر كتالي /dev/My_Spare_Disk .

```
KERNEL=="hdb" , DRIVER=="ide-disk" , SYMLINK+="spare_disk"
```

القاعده السابقه تقول : الجهاز الأتي من النواه و الذي يتمثل في مفتاح Match و الذي أسمه hdb الذي أسمه المُعلن ide-disk , سيكون أسمه الإفتراضي الذي سيُصبح عليه و الذي سيتم إنشاء وصله له بأسم spare_disk .
الملاحظة هنا أننا لم نحدد أسم الـ device node , لذالك فسيقوم udev بإستخدام الأسم الإفتراضي , من أجل الحفاظ على تخطيط مجلد /dev , فالقاعده الخاصه بك سيكون إذنها على الـ NAME فقط , لكن إنشاء Symlink أو أدي واجبات أخرى .

```
KERNEL =="hdc", SYMLINK +="cdrom cdrom0"
```

القاعده السابقه قد تكون أكثر القواعد نموذجيه عن سابقها من القواعد , فقد تم إنشاء إختصاربن Symbolic Link بأسماء /dev/cdrom و /dev/cdrom0 وكليهما لهم نقطة واحدة وهي /dev/hdc , ولم نستخدم الخاصية NAME لأن الأسم الإفتراضي التي تستخدمة النواه هو hdc .

<u>ملائمة صفات Sysfs</u>

حتى الأن الـ MatchKey التي قمنا بتقديمها تقوم بقدرات محدوده جداً , و المطلوب أن نقوم بعمل سيطره أكثر من هذا , فنحن نريد تحديد الأجهزة على أساس الخصائص المتقدمة مثلاً على أساس نظام لمجموعة قواعد و قوانين أو رقم المنتج أو الرقم المسلسل أو سعة التخزين للجهاز أو رقم القسم في القرص الصلب .. إلخ .

بعض الأجهزه تقوم بتصدير مثل هذه المعلومات إلى Sysfs , و udev يسمح لنا بدمج هذه المعلومات و توفيق العمليات في القاعده rules , عن طريق إستخدام المفتاح ATTR مع قليل من الإختلاف في صياغة القاعده rules .
الأن لدينا قاعدة الحكم الذي سينفذ فيها على أساس من معلومات من sysfs , وفيما بعد سنقوم بشرح المزيد عن طريقة كتابة قواعد مبنيه على الـ sysfs .

```
SUBSYSTEM=="block" , ATTR{size}=="2234343824334" , SYMLINK+="my_disk"
```

بحث تعليمي

النظام الهرمي للأجهزة

في نواة لينكس تُمثل الأجهزة في بنيه شبيه بالشجره , ومعلومات هذه الأجهزه تُكشف عن طريق sysfs حيث أنها تكون مفيده عند كتابتنا للقواعد , فعلى سبيل المثال , قرصي الصلب هو أبن لفرع SCSI Disk Device الذي هو فرع من أجهزة ATA , و الذي هو فرع من أجهزة الـ PCI bus (يُمكنك مُراجعة المخرجات التي في أول البحث) , و من المحتمل أن تجد نفسك بحاجة إلى معلومات من أحد الفروع عن طريق تقديم إستعلام أو سؤال , فمثلاً إذا كنت أريد الرقم المسلسل للقرص الصلب , فبكل بساطه سنقوم بسؤال الفرع الأكبر منه أو الأب للقرص الصلب الذي هو SCSI Disk .

وهناك مفاتيح match-key هام لدينا وسيُضاف إلى المجموعه السابق ذكرها وهي (ATTR) , قيم الـ match-key تصلح فقط للأجهزه ولكن لا تصلح للفروع الأساسيه , ولكن udev يقدم متغيرات لـ match-key تُمكنه من البحث و التنقيب في المستويات العليا أو الفروع العليا للجهاز .

وبهذا سنقوم بإعادات شرح جميع الـ Match-Key السابقة في حالة الإستعلام .

KERNEL	الأسم الممنوح من النواه للجهاز , أو الأسم الممنوح من النوه للفرع
SUBSYSTEM	النظام الفرعي للجهاز , أو الأسم الفرعي لأي فرع للأي جهاز
DRIVER	أسم التعريف الذي يدل على الجهاز , أسم التعريف الذي يدل على أي فرع لجهاز
ATTR	صفة sysfs للجهاز , أو صفة sysfs لأي فرع لجهاز

أعتقد بعدما إتضحة فكره النظام الهرمي للأجهزه أن كتابة القواعد الأن ستكون أقل تعقيداً من السابق , و لا داعي للقلق حيث أن هناك أدوات تقوم بمُساعدتك , وسيتم عرضها عليك في مرحلة مُتقدمه من هذه الوثيقه .

سلسلة الإستبدالات
String substitutions

عند كتابة القواعد التي من المحتمل تعاملها مع العديد من الأجهزه , فيكون من المستحسن إستخدام صيغ مُختصره مفهومه لـ udev , و أشهر إختصارين يُمكن الإستفاده منهم في هذه الحالة هم الإختصارين %k و %n .

%k : تُعبر عن أسم الجهاز داخل النواه , مثل "sda1" ويكون المكان الإفتراضي له هو /dev/sda1 .
%n : تُعبر عن رقم الجهاز في النواه أو رقم القسم للقرص الصلب , فمثلاً رقم 1 لـ sda1 وهكذا .

ولفهم طريقة إستخدام أسلوب String Substitutions تابع المثال التالي

```
KERNEL=="mice", NAME="input/%k"
KERNEL=="loop0", NAME="loop/%n", SYMLINK+="%k"
```

أول قاعده تقول : إنقل الجهاز الموجود في النواه بأسم mice إلى المجلد التالي /input حتى يكون كتالي /dev/input/mice

أما القاعده الثانيه تقول : إنقل الجهاز loop0 الذي في النواه إلى مجلد /loop حتى يكون المسار الخاص به كتالي /dev/loop/0 و قم بإنشاء إختصار له دائم في المجلد الرئيسي حتى يكون بهذا الشكل /dev/loop0 .

لكن في هذه القاعدتين السابقتين شك , لأنها كلها يمكن أن تعاد صياغتها دون استخدام أسلوب String Substitutions , لكن فائدتها الجوهريه ستتضح في الجزء التالي .

String Matching

عندما يتأكد udev من كتابة الـ Matching String بأسلوب صحيح , فسيسمح لك بإستخدام بعض العلامات (شبيه بعمليات Shell) , و هم ثلاث علامات كتالي :

بحث تعليمي

* : تعني أي حرف , أو صفر أو أي علامه
? : تعني تخمين حرف واحد فقط
[ ] : تعني تخمين أي حرف أو رقم من داخل مجال القوسين

المثال التالي يوضح كيفية إستخدام هذه العلامات مع الجزء السابق String Substitutions .

```
KERNEL=="fd[0-9]*", NAME="floppy/%n", SYMLINK+="%k"
KERNEL=="hiddev*", NAME="usb/%k"
```

في القاعده الأول نقول فيها : إبحث عن كل أجهزة الـ floppy disk و الذي ستعثر عليه قم بإضافته في مجلد floppy لكي يكون المسار كتالي /dev/floppy/0 و قم بإنشاء إختصار له في المجلد الرئيسي لكي يكون كتالي /dev/fd0 (بأسم الجهاز الصادر من النوه كما هو) .

و نُلاحظ أن في المفتاح الأول قمنا بتحديد مدى الأرقام التي سيبحث فيها وهي من 0 إلى 9 و قمنا بإحاطتهم بالقوسين المربعين [ ] , وألحقناها بالنجمه التي تعني أي شيء بعده إذا وُجد ذالك .

في القاعده الثانيه نقول فيها : إيبحث عن أي جهاز يبدء بـ hiddev و قم بإضافته في مجلد usb حتي يكون مساره كتالي /dev/usb/hiddev4392 .

بحث تعليمي

العثور على المعلومات من sysfs
شجرة Sysfs

فكرة الإعتماد على معلومات من sysfs فكره رائعه و مُمتعه جداً , و بالطبع لفهم هذه الفكره بصوره واعيه , يجب عليك أنت تكون قد إستوعبت ما ذكرناه سابقاً , وللبدء في كتابة rules مُعتمده على المعلومات , ييجب أولاً معرفة بعض أسماء الخصائص و القيم المتداوله لها .

Sysfs في الواقع له بُنيه شديدة البساطه , فهو ينقسم بصوره منطقيه إلى مجموعة دلائل , كل دليل يحتوي على رقم attributes أي يحتوي على قيمه واحده فقط , وهناك بعض الإختصارات موجوده , التي تتصل بالفرع الرئيسي لها (ستفهم معناها فيما بعد) والذي سيُمثل الشكل الهرمي كما ذكرنا لاحقاً .

بعض هذه الأدلة الموجوده يُشار بها بـ top-level device path , هذا الأدلة تماماً مثل الأجهزه الحقيقيه ولها أيضاً device node تتعامل معه , وهي تصنف في دليل الـ sysfs بأن لها أجهزه في المجلد dev , و الدليل على هذا الأمر التالي

```
find /sys -name dev
```

على سبيل المثال : لدي على جهازي المسار التالي /sys/block/sda وهو مسار القرص الصلب لدي و يُسمى (device path) , وهو متصل بالفرع الرئيسي له و هو SCSI disk device , وبإمكانك مشاهده هذا عن طريق وجود إختصار بأسم device , أي المسار كتالي

/sys/block/sda/device

ستكتشف أنه الفرع الرئيسي لقرصي الصلب .

وعندما تحب أن تكتب قاعدة مبنيه على معلومات من sysfs , فبكل سهوله ستضع الصفه التي تُريد من أي جزء من صفات جهازك .
بصوره أوضح , بإمكاني إستخدام مساحة قرصي الصلب كصفة , وبإمكاني معرفة ذالك عن طريق التالي

```
cat /sys/block/sda/size
234441648
```

عند إستخدام هذه المعلومه كمعرف للقرص الصلب sda مع قواعد الـ udev ستكون كتالي .

```
ATTR{size}=="234441648"
```

عندها سيقوم udev بالكشف عن هذه السعه في كل الأجهزه الموجوده عنده (أقصد الأقراص الصلبه) , و من الأفضل إختيار صفات أخرى إذا أمكن هذا فمثلاً بإمكانك التجول في /sys/block/sda/device و معرفة محتويات الملفات الموجوده وهذا بالطبع عن طريق الأمر cat كما شرحنا في المثال السابق , وجميعهم سيكون معرفهم في ملف القواد بـ ATTR كما شرحنا كما أن هناك سلسله ملفات (أو مجموعة ملفات) يجب الحذر في التعامل معها وسوف نستعرضها فيما بعد .

وعلى الرغم من أن هذه مقدمه رائعه لمعرفة كيفية الحصول على معلومات من sysfs و أيضاً معرفة الهيكل الخاص به و طريقة تعاملنا معه من خلال الـ udev , إلى أن كل هذا مضيعه للوقت فإذا زاد العمل معها زاد العناء .

Udevinfo OR udevadm

ملاحظة :
إذا لم يكن متوفر لديك برنامج udevinfo فبإمكانك إستبداله ببرنامج udevadm

هذه البرامج سهله جدا و بسيطه تُساعدك على كتابه القاعده الخاصه بك بصوره إحترافيه , وكل ما تحتاج إليه هو معرفة مسار جهازك داخل بُنيةُ sysfs (أي داخل مجلد sys) و شاهد المثال التالي :

udevadm

```
# udevadm info -a -p /sys/block/sda
```

**Udevadm info starts with the device specified by the devpath and then walks up the chain of parent devices. It prints for every device found, all possible attributes in the udev rules key format.
A rule to match, can be composed by the attributes of the device and the attributes from one single parent device.**

```
  looking at device '/block/sda':
    KERNEL=="sda"
    SUBSYSTEM=="block"
    DRIVER==""
    ATTR{range}=="16"
    ATTR{ext_range}=="256"
    ATTR{removable}=="0"
    ATTR{ro}=="0"
    ATTR{size}=="234441648"
    ATTR{capability}=="52"
    ATTR{stat}=="   77280     5163  2725213   667503    47734   125055  1382528
629930        0   392870  1297360"

  looking at parent device
'/devices/pci0000:00/0000:00:1f.2/host2/target2:0:0/2:0:0:0':
    KERNELS=="2:0:0:0"
    SUBSYSTEMS=="scsi"
    DRIVERS=="sd"
    ATTRS{device_blocked}=="0"
    ATTRS{type}=="0"
    ATTRS{scsi_level}=="6"
    ATTRS{vendor}=="ATA     "
    ATTRS{model}=="Hitachi HTS54161"
    ATTRS{rev}=="SBDO"
    ATTRS{state}=="running"
    ATTRS{timeout}=="30"
    ATTRS{iocounterbits}=="32"
    ATTRS{iorequest_cnt}=="0x1e86e"
    ATTRS{iodone_cnt}=="0x1e86e"
    ATTRS{ioerr_cnt}=="0x1"
    ATTRS{modalias}=="scsi:t-0x00"
    ATTRS{evt_media_change}=="0"
    ATTRS{queue_depth}=="31"
    ATTRS{queue_type}=="simple"

  looking at parent device '/devices/pci0000:00/0000:00:1f.2/host2/target2:0:0':
    KERNELS=="target2:0:0"
    SUBSYSTEMS==""
    DRIVERS==""

  looking at parent device '/devices/pci0000:00/0000:00:1f.2/host2':
    KERNELS=="host2"
    SUBSYSTEMS==""
    DRIVERS==""

  looking at parent device '/devices/pci0000:00/0000:00:1f.2':
    KERNELS=="0000:00:1f.2"
    SUBSYSTEMS=="pci"
```

```
    DRIVERS=="ahci"
    ATTRS{vendor}=="0x8086"
    ATTRS{device}=="0x27c5"
    ATTRS{subsystem_vendor}=="0x1558"
    ATTRS{subsystem_device}=="0x0660"
    ATTRS{class}=="0x010601"
    ATTRS{irq}=="28"
    ATTRS{local_cpus}=="ff"
    ATTRS{local_cpulist}=="0-7"
    ATTRS{modalias}=="pci:v00008086d000027C5sv00001558sd00000660bc01sc06i01"
    ATTRS{enable}=="1"
    ATTRS{broken_parity_status}=="0"
    ATTRS{msi_bus}==""

  looking at parent device '/devices/pci0000:00':
    KERNELS=="pci0000:00"
    SUBSYSTEMS==""
    DRIVERS==""
```

udevinfo

```
# udevinfo -a -p /sys/block/sda

  looking at device '/block/sda':
    KERNEL=="sda"
    SUBSYSTEM=="block"
    ATTR{stat}=="  128535     2246  2788977   766188    73998   317300  3132216
5735004        0   516516  6503316"
    ATTR{size}=="234441648"
    ATTR{removable}=="0"
    ATTR{range}=="16"
    ATTR{dev}=="8:0"

  looking at parent device
'/devices/pci0000:00/0000:00:07.0/host0/target0:0:0/0:0:0:0':
    KERNELS=="0:0:0:0"
    SUBSYSTEMS=="scsi"
    DRIVERS=="sd"
    ATTRS{ioerr_cnt}=="0x0"
    ATTRS{iodone_cnt}=="0x31737"
    ATTRS{iorequest_cnt}=="0x31737"
    ATTRS{iocounterbits}=="32"
    ATTRS{timeout}=="30"
    ATTRS{state}=="running"
    ATTRS{rev}=="3.42"
    ATTRS{model}=="ST3120827AS"
    ATTRS{vendor}=="ATA"
    ATTRS{scsi_level}=="6"
    ATTRS{type}=="0"
    ATTRS{queue_type}=="none"
    ATTRS{queue_depth}=="1"
    ATTRS{device_blocked}=="0"

  looking at parent device '/devices/pci0000:00/0000:00:07.0':
    KERNELS=="0000:00:07.0"
    SUBSYSTEMS=="pci"
```

```
DRIVERS=="sata_nv"
ATTRS{vendor}=="0x10de"
ATTRS{device}=="0x037f"
```

كما رأيتم فكل من البرنامجين يُقدم مجموعه من الـ match-key و الـ attributes-key التي بإمكانك إستخدامها في ملف القواعد الخاص بك , فبإمكاني الإستفاده من المعلومات السابقه كتالي مثلاً

```
SUBSYSTEM=="block", ATTR{size}=="234441648", NAME="my_hard_disk"
SUBSYSTEM=="block", SUBSYSTEMS=="scsi", ATTRS{model}=="ST3120827AS",
NAME="my_hard_disk"
```

المكتوب باللون الأزرق , هو حصر للقرص الصلب (على إعتبار أن هناك أكثر من قرص واحد) , و أيضاً لمعرفة انه من الوارد إستخدام أكثر من معلومه في القاعده الواحده , ولكن ليس من القانوني أن تقوم بالمزج بين الـ attributes-key في الأجزاء الرئيسيه , أنظر المثل التالي لكي تفهم ما أقصده جيداً و بعد ذالك قارن المثال التالي بالمثال السابق و لاحظ مخرجات البرنامج .

```
SUBSYSTEM=="block", ATTRS{model}=="ST3120827AS", DRIVERS=="sata_nv",
NAME="my_hard_disk"
```

هكذا لن تعمل هذه القاعدة و سيتم رفضها من udev , وعلى أي حال , يقوم البرنامج بتزويدك بعدد ضخم من الـ attributes-key وبإمكانك إختيار أي عدد منهم لتركيب قاعدتك دون اللجوء إلى التشتت بين الفروع الرئيسيه , ويجب أيضاً أن تختار attributes-key يدل على الجهاز بصفه دائمه (اي لا تختار الأصفار 0) , أي أنني عندما أخترت في المثال الأول attributes كان يدل على مساحة القرص , ولم استخدم أرقام غامضة مثل
ATTR{iodone_cnt}="0x2324}

لاحظ تدرج المُخرجات في برنامج udevadm أو udevinfo , في اللون الأزرق , يُمكن عمل إستعلام أي بإستخدام الـ Match-key مثل ATTR , KERNEL.

في الون الأخضر الأحمر الداكن نستعلم من خلال الفروع الرئيسيه عن طريق الـ attributes-key مثل SUBSYSTEMS,ATTR .

قد يكون هناك سؤال الأن يدور في عقلك وهو , كيف أحدد الجهاز المطلوب في مجلد sys ؟

بإمكانك تحديد لأحد البرنامجين المسار المطلوب للجهاز إذا كنت تعرفه أو تدعه هو الذي يبحث لك عنه , وهذا عن طريق المثال التالي

udevadm

```
udevadm info -a -p $(udevadm info -q path -n /dev/sda)
```

udevinfo

```
udevinfo -a -p $(udevinfo -q path -n /dev/sda)
```

الطرق البديله

على الرغم من أن udevadm أو udevinfo يُقدمان أسلوب بسيط للحصول على المعلومات الخاصه بالجهاز و التي تُساعدك في كتابة قواعدك , إلى أن هناك برامج أفضل منها مثل usbview والتي تقوم أيضاً بإظهار الكثير من المعلومات .

بحث تعليمي

المواضيع المُتقدمة

التحكم في الترخيص(Permissions) و الملكيات (ownership)

يسمح لك udev بإضافة assignment-key أخرى تقوم بدورها بالتحكم في التراخيص و الملكيات على كل الأجهزه . فالمفتاح GROUP يسمح لك أن تُحدد أي مجموعه ينتمي إليها device node ما في القاعد , و المثال التالي يُوضح كيف سنقوم بمنح جهاز framebuffer إلى مجموعة الـ Video

```
KERNEL==”fb[0-9]*”, NAME=”fd%n”, SYMLINK+=”%k” , GROUP=”video”
```

المفتاح OWNER ربما يكون أقل فائده ولكنه يسمح بأن تقوم بتحديد أي مستخدم هو الذي له السلطه و الصلاحيه على device node ما , على إفتراض أن هذا الجهاز حاله إستثنائيه أو حكر على مستخدم مُعين , فمثلاً إذا كنت أريد أن أجعل قرصي الخارجي حكر على المستخدم Abaza فقط فسيكون كتالي

```
KERNEL==”sdb[0-9]*”, OWNER=”Abaza”
```

udev يقوم إفتراضياً عند إنشاء أي جهاز بمنح ترخيص 0660 (قراءه و كتابه) للمستخدم و مجموعته , فمثلاً إذا أردنا تغير ترخيص لجهاز ما , ولنفترض أنه جهاز الأقراص المرنه (Floppy Disk) , فسنستخدم المفتاح MODE لهذا الغرض كما بالمثال التالي

```
KERNEL==”fd[0-9]*”, NAME=”floppy/%n”,SYMLINK+=”%k” , MODE=”0666”
```

تم هكذا إعطاء تصريح للقراء و الكتابه للجميع (غير مستحسن عمل هذا عشوائياً إلى إذا كنت تعرف ما تفعل).

أستخدام برنامج خارجي لتسمية الأجهزه

في بعض الأحيان قد نحتاج إلى برنامج يقوم هو بإختيار أسماء للأجهزه و غير المألوفه بالنسبه للنواه و udev , ولكن udev يُقدم إمكانيه لطلب تشغيل برنامج عند العثور على جهاز مُعين , و إعطائه الأسم الأصلي لكي يقوم البرنامج بإخراج الأسم الجديد.

و لإستخدام هذه الطريقه , أولاً نحتاج إلى وجود برنامج يهتم بهذا (أي أنه من صنعك مثلاً) , و سأفترض أن هناك برنامج يقوم بهذه المهمه و مكانه في /usr/local/bin و أسمه ch_dev_name , بإستخدام مفتاح PROGRAM في القاعده و إعطائه الأسم المطلوب تغيره , لكي يقوم البرنامج بإعطاء الناتج في المتغير %c و الذي يُمكن إستخدامه مع المفتاحين NAME , SYMLINK , ستكون القاعده كتالي

```
KERNEL==”hda”, PROGRAM=”/usr/local/bin/ch_dev_name %k”, SYMLINK+=”%c”
```

المثال السابق قام البرنامج بأخذ بارميتر صادر من النواه و الذي يُمثل أسم الجهاز في النواه , و بناء على هذا سيقوم ch_dev_name بإنتاج أسم جديد كمخرج عادي stdout , ينقسم إلى عدة أجزاء , كل جزء عباره عن كلمه واحده يُفصل بينهما بمسافه (مثل المصفوفات في البرمجه).

و أفترضنا أن البرنامج أخرج أرقام الأجزاء التي قمنا بإعطائها للمفتاح SYMLINK لإنشائها بدلاً من الأسماء.

في المثال التالي سنفترض أن البرنامج ch_dev_name أخرج جزئين , الأول لأسم الجهاز و الثاني لأسم الإختصار , في هذا الحاله سنقوم بإستخدام العلامه السابقه مع القليل من الإضافات كي تكون بهذا الشكل c%{N} حيث أن N هو رقم الجزء , وتكون القاعده كتالي

```
KERNEL==”hda”, PROGRAM=”/usr/local/bin/ch_dev_name %k”, NAME=”%c{1}” ,
SYMLINL+=”%c{2}”
```

نفترض أن البرنامج يقوم بإخراج أكثر من جزء , لحل هذا سيكون العلامه بهذا الشكل c%{N+} حيث أن N+ هي الجزء

بحث تعليمي

المطلوب و تكون ترتيبها كتالي N,N+1,N+2,..etc لأخر المخرجات , تابع التعديل على المثال السابق.

```
KERNEL=="hda", PROGRAM="/usr/local/bin/ch_dev_name %k", NAME="%c{1}" ,
SYMLINL+="%c{2+}"
```

من الممكن إستخدام هذه الطريقه مع مفاتيح أخرى ليس فقط مع NAME و SYMLINK , تابع هذا المثال و نحن نقوم بجعل برنامج لنا يقوم بإعطاء التصريح للمجموعه التي يرى البرنامج أنها مُناسبه له .

```
KERNEL=="hda", PROGRAM="/usr/local/bin/chdevgroup %k" , GROUP="%c"
```

تشغيل برنامج على حدث مُعين

هناك دافع أخرى لكتابة قواعد تُمكننا من تشغيل برنامج في حالة إتصال جهاز مُعين و فصله , فمثلاً تخيل أنك تُريد أن تكتب برنامج يقوم بتحميل كل الصور في مجلد معين عندما تقوم بتوصيل الكميرا الديجيتل أو الهاتف الخلوي .

قد يأتي في مُخيلتك أن المفتاح PROGRAM الذي تحدثنا عنه منذ قليل يفي بالغرض , ولكن الإجابه بلا , لأن وظيفة المفتاح PROGRAM هي إنشاء برنامج لأسماء للأجهزه فقط لا غير , لأن هذا المفتاح يقوم بتنفيذ البرنامج **قبل** أن يتم إنشاء الجهاز , لذالك فهذا المفتاح لا يجوز إستخدامه في فكرتنا .

وظيفة المفتاح الذي سنتحدث عنه الأن تسمح لك بتنفيذ البرنامج المحدد **بعد** إنشاء الجهاز و وضعه في مكانه , فهذا البرنامج سيعمل على الجهاز المحدد له , ولكن يجب أن تعلم أمر هام وهو أنه يجب أن تجعل البرنامج يتوقف بعد إتمام عمله , كما أنه يجب أن تجعل البرنامج يتأكد من أن الملفات تتطابق أم لا حتى تتجنب بعض الأخطأ التي من الوارد الوقوع بها .

المثال التالي يُوضح كيفية إستخدام المفتاح الذي يقوم بهذا العمل و هو المفتاح RUN .

```
KERNEL=="sdb" , RUN+="PROGRAM_PATH"
```

حيث أنك ستستبدل كلمة **PROGRM_PATH** بمسار البرنامج الخاص بك , فعند العمل سيكون هناك متغيرات من بيئة udev متوفره في بيئة المتغيرات الخاصه بالنظام , بما في ذالك قيم لمفاتيح مثل SUBSYSTEM , ويُمكنك الإستفاده منها عن طريق المفتاح ACTION لإكتشاف أي الأجهزه المتصله و الغير متصله , أي ستحتوي إما على add او remove .

لا يقوم udev بتشغيل البرنامج على أي نوع طرفيه , ولا حتى يتم كتابة نص سكربت في داخل القاعده , يقوم فقط بتنفيذ البرامج التي تبدء في مقدمة الملف بـ

```
#!/bin/sh
```

وخلاف ذالك لا يقوم بتنفيذ أي شيء , كما انه أيضاً لن يقوم بكتابة أي رساله في الطرفيه .

البيئة التفاعليه

udev توفر مفتاح ENV الذي يوفر البيئه تفاعليه التي يُمكن إستخدامها سواء مع مفاتيح Match و مفاتيح Assignment ,

في حالة الـ Assignment فيُمكنك تعين متغير تستفيد منه فيما بعد مع مفاتيح Match , كما أنه بإمكانك إستخدام بيئة المتغيرات مع أحد البرامج التنفيذيه المذكوره سابقاً بأسلوب معين معها , أنظر المثال التالي لمعرفة طريقة إستخدام المفتاح ENV

```
KERNEL=="fd0" , SYMLINK+="floppy" , ENV{some_env}="VALUE"
```

في حالة الـ `Match` , بإمكانك أن تجعل القاعده تعتمد على متغير معين في بيئة المتغيرات , مع ملاحظة أنه يجب أن يكون المتغير متوفر في بيئة المتغيرات و يُمكن للمستخدم معرفة قيمته من خلال الـ `Console` , تابع المثال التالي

```
KERNEL=="fd0" , ENV{some_env}=="yes" , SYMLINK+="floppy"
```

كأنك تستخدم قاعدة `if` في البرمجه , و القاعده السابقه تقول أنه إذا توفر المتغير `some_env` ويحمل قيمه `yes` قم بعمل

بحث تعليمي

إختصار للجهاز fd0 في المجلد dev بأسم floppy

خيارات إضافيه

هناك مفتاح يُمكنك إستخدامه على حسب فكرتك الشخصيه , المفتاح من نوع assignment و أسمه OPTIONS و هذا المفتاح يقبل ثلاث قيم فقط كالتي

all_partions	إنشاء كل الأقسام الممكن إنشائها في الـ device Block, وليس فقط التي تم الكشف عنها عند الإقلاع
ignore_device	تجاهل الحدث تماماً
last_rule	عدم تنفيذ لأي قواعد أخرى

المثال التالي يقوم بجعل القرص الصلب ينتمي إلى مجموعة الأقراص الصلبه و بعد ذالك لا يقوم بتنفيذ أي قاعده أخرى تتعلق به

```
KERNEL=="sda", GROUP="disk" , OPTIONS+="last_rule"
```

بعض الأمثله

USB Printer

لنفرض أن لدينا طباعه , وقمنا بتوصيله بالجهاز , فأنتج لدينا مسار /dev/lp0 , ونريد الأن تغير أسمه إلى أسم أكثر مدلوليه , سنقوم بالإعتماد على برنامج udevadm الذي سيقوم بتزويدنا بالقواعد التي سنكتبها .

```
udevadm info -a -p$(udevadm info -q path -n /dev/lp0)
looking at device '/class/usb/lp0':
    KERNEL=="lp0"
    SUBSYSTEM=="usb"
    DRIVER==""
    ATTR{dev}=="180:0"

  looking at parent device '/devices/pci0000:00/0000:00:1d.0/usb1/1-1':
    SUBSYSTEMS=="usb"
    ATTRS{manufacturer}=="EPSON"
    ATTRS{product}=="USB Printer"
    ATTRS{serial}=="L72010011070626380"
```

إذا ستكون القاعده كتالي :

```
SUBSYSTEM=="usb", ATTRS{serial}=="L72010011070626380", SYMLINK+="epson_680
```

لكي يكون أسم الجهاز بعد ذالك ومساره /dev/epson_680

USB Camera

الكمرا الرقميه الخاصه بي تُعرف على أنها قرص خارجي عندما أقوم بتوصيلها في منفذ الـ USB , و تستخدم SCSI transport , ويقوم النظام بضمها و أقوم أنا بأخذ الصور من الكميرا و أحفظها في المكان المخصص للصور على القرص الصلب الرئيسي , ولكن ليس كل الكميرات تعمل بهذه الطريقه , فبعض الكميرات الرقميه وخصوصاً القديمه لا تقوم بإنشاء منفذ لقرص خارجي ولكن هناك مكتبه أسمها libgphoto2 أو gphoto2 تقوم بعلاج هذه المشكله , وفي هذه الحاله لا يكون هناك أي حاجه إلى كتابة قواعد لهذا الكميرا .
المشكله الرئيسيه مع الكميرات الرقميه أنها تقوم بإنشاء قرص خارجي واحد فقط (أي ليس مُقسم 1 و 2 و..) فمثلاً إذا كان لدي sdb و sdb1 فعند اذا sdb1 افيد لي من sdb لأن sdb1 هو الذي أريد ضمه فقط , و الأكثر من هذا أنه يُقيد إمكانيات sysfs حيث أنه من المفيد للـ udevadm ان يقوم بتحديد المفاتيح المناسبه لـ sdb1 التي ستكون قسم في sdb , ولكن في حالتنا هذه ستكون النتيجه على إثنين أي القرص الرئيسي و الفرعي , و هذا الذي لا نريده , لذالك يجب أن تكون القاعده شيء

بحث تعليمي

خاص جداً .

و سيكون من المدهش إذا عرفت ما الفرق بين الأسمين sdb و sdb1 ... الفرق في الأسم فقط (في حالة الكميرا) , أي أن بإمكانك أستخدامها كمفتاح Match و إنشاء sdb1 عن طريق المفتاح NAME . و ببساطه سنقوم بكتابة التالي للحصول على المطلوب .

```
# udevadm info -a -p $( udevadm info -q path -n /dev/sdb1)
  looking at device '/block/sdb/sdb1':
    KERNEL=="sdb1"
    SUBSYSTEM=="block"

  looking at parent device '/devices/pci0000:00/0000:00:02.1/usb1/1-1/1-
1:1.0/host6/target6:0:0/6:0:0:0':
    KERNELS=="6:0:0:0"
    SUBSYSTEMS=="scsi"
    DRIVERS=="sd"
    ATTRS{rev}=="1.00"
    ATTRS{model}=="X250,D560Z,C350Z"
    ATTRS{vendor}=="OLYMPUS "
    ATTRS{scsi_level}=="3"
    ATTRS{type}=="0"
```

وسنكتب القاعده كتالي

```
KERNEL=="sd?1", SUBSYSTEMS=="scsi", ATTRS{model}=="X250,D560Z,C350Z",
SYMLINK+="camera"
```

تفسير علامة (?) هو أننا لا نستطيع تحديد القاعده بأنها دائماً ستكون sdb فإفترض وضعك لقرص خارجي

USB Hard Disk

مثل فكرة الكميرا هي نفس فكره القرص الصلب الخارجي ولكن من الممكن تقسيم القرص الصلب عن طريق الأمر fdisk بينما لا يُمكن تقسيم الكميرا !!

ولكن قاعدة القرص الصلب الخارجي سهله جده و سنفترض الإحتماليين إذا كان مُقسم إلى أكثر من قسم أم لا

```
KERNEL=="sd*", SUBSYSTEMS=="scsi", ATTRS{model}=="USB 2.0 Storage Device",
SYMLINK+="usbhd%n"
```

نتائج هذه القاعده ستكون كتالي

/dev/usbhd
/dev/usbhd1
/dev/usbhd2

بحث تعليمي

USB Card Reader

هناك أنواع كثيره من أنواع قارة كروت الذاكره مثل CompactFlash, SmartMedia و غيره الكثر و هي تعتبر أيضاً من وسائل التخزين الخارجيه مثل غيرها .

لكن نفرض أنك قمت بوضع جهاز القارء نفسه في الجهاز مع كرت ذاكره فسيكون هناك قسم و قسم فرعي أي sdb و sdb1 ولكن في حالة إنتزاع الذاكره فبذالك لن يكون هناك sdb1 لأن الجهاز لن يُدرك هذا وهذا قد يُسبب بعض المشاكل خصوصاً عند إضافه قرص خارجي أخر و الكارثه تكون عندما يكون القرص الخارجي هام .. فهناك فرصه للخطأ كبيره من جهة المستخدم سواء في المسح او التهيئه أو غيره من العمليات التي قد تُسبب فقد للمعلومات الهامه , المشكله الأخرى و هو أننا نفترض أن كرت الذاكره مُقسم للعديد من الاقسام , فستكون الكارثه أكبر و أكبر .

ولكن لعلاج هذه المشكله سنستخدم في القاعد المفتاح OPTIONS مع الخيار all_partions وستكون القاعده كتالي

```
KERNEL="sd*", SUBSYSTEMS=="scsi", ATTRS{model}=="USB 2.0 CompactFlash Reader",
SYMLINK+="cfrdr%n", OPTIONS+="all_partitions"
```

سيكون نتائج القاعده كتالي

/dev/cfrdr, cfrdr1, cfrdr2, cfrdr3, …

Network Interface

على الرغم من أن الشبكه لها أسم معروف إلى أنها لا تمتلك Device Node في مجلد dev , إلى أن كتابة قاعدة لها أمر سهل تنفيذه .

كل ما سنحتاج إليه لتميز جهاز الإتصال بالشبكه هو الـ MAC حيث أنه رقم مُميز يُمكن الإعتماد عليه ولكن تأكد جيداً أنه مكتوب بصوره صحيحه في القاعده وإلى لن يقوم udev بتنفيذها .

```
# udevadm info -a -p /sys/class/net/eth0
  looking at device '/class/net/eth0':
    KERNEL=="eth0"
    SUBSYSTEM=="net"
    DRIVER==""
    ATTR{addr_len}=="6"
    ATTR{dev_id}=="0x0"
    ATTR{ifalias}==""
    ATTR{iflink}=="2"
    ATTR{ifindex}=="2"
    ATTR{features}=="0x180"
    ATTR{type}=="1"
    ATTR{link_mode}=="0"
    ATTR{address}=="00:90:f5:53:9d:c2"
    ATTR{broadcast}=="ff:ff:ff:ff:ff:ff"
    ATTR{carrier}=="1"
    ATTR{dormant}=="0"
    ATTR{operstate}=="up"
    ATTR{mtu}=="1500"
    ATTR{flags}=="0x1003"
    ATTR{tx_queue_len}=="1000"

  looking at parent device '/devices/pci0000:00/0000:00:1e.0/0000:05:00.0':
    KERNELS=="0000:05:00.0"
    SUBSYSTEMS=="pci"
```

```
    DRIVERS=="r8169"
    ATTRS{vendor}=="0x10ec"
    ATTRS{device}=="0x8169"
    ATTRS{subsystem_vendor}=="0x1558"
    ATTRS{subsystem_device}=="0x0660"
    ATTRS{class}=="0x020000"
    ATTRS{irq}=="18"
    ATTRS{local_cpus}=="ff"
    ATTRS{local_cpulist}=="0-7"
    ATTRS{modalias}=="pci:v000010ECd00008169sv00001558sd00000660bc02sc00i00"
    ATTRS{broken_parity_status}=="0"
    ATTRS{msi_bus}==""

  looking at parent device '/devices/pci0000:00/0000:00:1e.0':
    KERNELS=="0000:00:1e.0"
    SUBSYSTEMS=="pci"
    DRIVERS==""
    ATTRS{vendor}=="0x8086"
    ATTRS{device}=="0x2448"
    ATTRS{subsystem_vendor}=="0x0000"
    ATTRS{subsystem_device}=="0x0000"
    ATTRS{class}=="0x060401"
    ATTRS{irq}=="0"
    ATTRS{local_cpus}=="ff"
    ATTRS{local_cpulist}=="0-7"
    ATTRS{modalias}=="pci:v00008086d00002448sv00000000sd00000000bc06sc04i01"
    ATTRS{broken_parity_status}=="0"
    ATTRS{msi_bus}=="1"

  looking at parent device '/devices/pci0000:00':
    KERNELS=="pci0000:00"
    SUBSYSTEMS==""
    DRIVERS==""
```

إذا نظرت على المخرجات جميعها أرقام غامضه ولكن الشيء الوحيد الذي يُمكن أن نثق فيه هو البيان address و بناء على هذا ستكون القاعده كتالي

```
KERNEL=="eth*",ATTR{address}=="00:90:f5:53:9d:c2",NAME="lan"
```

ستحتاج إلى إعادة تشغيل الشبكه للحصول على النتائج الجديده , وبإمكانك أيضاً إعادة تشغيل الوحدات التي تعمل مع الشبكه و الأفضل من كل هذا إعادة تشغيل النظام عن طريق المفتاح SysRq (هناك شرح في مجتمع لينكس العربي) , وقد تحتاج إلى تعديل الإعدادات الخاص بالشبكه لإستبدال eth بـ lan (من المستحسن تجربة تغير الإعدادات قبل إنشاء القاعده حيث أن هناك توزيعات ترفض التغير) بعد هذا بإمكانك إستخدام الأسم الجديد في كل البرامج مثل ifconfig و غيرها من البرامج بدلاً من الأسم القديم .

بحث تعليمي

الإختبار و التنقيح

وضع القواعد على وضع العمل

تقوم udev بإحضار ملفات القواعد و تشغيلها تلقائياً كل هذا بالإعتماد على جزء في النواه يُسمى inotify , أي أنه إذا قمت مثلاً بإنشاء إختصار SYMLINK للكميرا , فسيقوم udev بالتعرف على التغير الجديد بمجرد فصل الكميرا و توصيلها مره أخرى , أو بإمكان بدلاً من هذا كتابة الأمر التالي بدلاً من الفصل و التوصيل للجهاز

```
udevadm trigger
```

أما إذا كانت النواه لا تدعم inotify فلن يقوم udev بضم القواعد الجديده تلقائياً , وفي هذه الحاله ستقوم بكتابة الأمر التالي بعد كتابة كل قاعده جديده و بعد كل تعديل على قاعده سابقه

```
udevadm control --reload_rules
```

إذا كنت تعرف الفرع الرئيسي (Top-level) لمسار جهاز ما في الـ sysfs , فبإمكانك إستخدام الأمر udevadm test لمعرفة الأحداث الذي يقوم بها udev مع هذا الجهاز , هذا الطريقه تُساعدك كثيراً في تنقيح قواعدك التي تكتبها , على إفتراض أنك تُريد الكشف على الجهاز الذي في المسار التالي /sys/class/sound/dsp فسيكون الأمر كتالي

```
$ udevadm test /sys/class/sound/dsp
run_command: calling: test
udevadm_test: version 141
This program is for debugging only, it does not run any program,
specified by a RUN key. It may show incorrect results, because
some values may be different, or not available at a simulation run.

parse_file: reading '/etc/udev/rules.d/10-vboxdrv.rules' as rules file
parse_file: reading '/lib/udev/rules.d/40-alsa.rules' as rules file
parse_file: reading '/lib/udev/rules.d/40-hplip.rules' as rules file
parse_file: reading '/lib/udev/rules.d/40-ia64.rules' as rules file
parse_file: reading '/lib/udev/rules.d/40-infiniband.rules' as rules file
parse_file: reading '/lib/udev/rules.d/40-isdn.rules' as rules file
parse_file: reading '/lib/udev/rules.d/40-libpisock9.rules' as rules file
parse_file: reading '/lib/udev/rules.d/40-pilot-links.rules' as rules file
parse_file: reading '/lib/udev/rules.d/40-ppc.rules' as rules file
parse_file: reading '/lib/udev/rules.d/40-xserver-xorg-input-wacom.rules' as
rules file
parse_file: reading '/lib/udev/rules.d/40-zaptel.rules' as rules file
parse_file: reading '/lib/udev/rules.d/45-fuse.rules' as rules file
parse_file: reading '/lib/udev/rules.d/45-libmtp8.rules' as rules file
parse_file: reading '/lib/udev/rules.d/50-firmware.rules' as rules file
parse_file: reading '/lib/udev/rules.d/50-udev-default.rules' as rules file
parse_file: reading '/lib/udev/rules.d/60-cdrom_id.rules' as rules file
parse_file: reading '/lib/udev/rules.d/60-persistent-input.rules' as rules file
parse_file: reading '/lib/udev/rules.d/60-persistent-serial.rules' as rules file
parse_file: reading '/lib/udev/rules.d/60-persistent-storage-tape.rules' as rules
file
parse_file: reading '/lib/udev/rules.d/60-persistent-storage.rules' as rules file
parse_file: reading '/lib/udev/rules.d/60-persistent-v4l.rules' as rules file
parse_file: reading '/lib/udev/rules.d/61-option-modem-modeswitch.rules' as rules
file
parse_file: reading '/lib/udev/rules.d/61-persistent-storage-edd.rules' as rules
file
parse_file: reading '/lib/udev/rules.d/62-bluez-hid2hci.rules' as rules file
parse_file: reading '/lib/udev/rules.d/64-device-mapper.rules' as rules file
parse_file: reading '/lib/udev/rules.d/65-dmsetup.rules' as rules file
```

```
parse_file: reading '/lib/udev/rules.d/70-acl.rules' as rules file
parse_file: reading '/etc/udev/rules.d/70-persistent-cd.rules' as rules file
parse_file: reading '/etc/udev/rules.d/70-persistent-net.rules' as rules file
parse_file: reading '/lib/udev/rules.d/75-cd-aliases-generator.rules' as rules
file
parse_file: reading '/lib/udev/rules.d/75-persistent-net-generator.rules' as
rules file
parse_file: reading '/lib/udev/rules.d/77-nm-probe-modem-capabilities.rules' as
rules file
parse_file: reading '/lib/udev/rules.d/77-probe-modem-capabilities.rules' as
rules file
parse_file: reading '/lib/udev/rules.d/79-fstab_import.rules' as rules file
parse_file: reading '/lib/udev/rules.d/80-drivers.rules' as rules file
parse_file: reading '/lib/udev/rules.d/85-alsa-utils.rules' as rules file
parse_file: reading '/lib/udev/rules.d/85-brltty.rules' as rules file
parse_file: reading '/lib/udev/rules.d/85-hdparm.rules' as rules file
parse_file: reading '/lib/udev/rules.d/85-hplj10xx.rules' as rules file
parse_file: reading '/lib/udev/rules.d/85-hwclock.rules' as rules file
parse_file: reading '/lib/udev/rules.d/85-ifupdown.rules' as rules file
parse_file: reading '/lib/udev/rules.d/85-pcmcia.rules' as rules file
parse_file: reading '/lib/udev/rules.d/85-regulatory.rules' as rules file
parse_file: reading '/etc/udev/rules.d/9-myrules.rules' as rules file
parse_file: reading '/lib/udev/rules.d/90-hal.rules' as rules file
parse_file: reading '/etc/udev/rules.d/95-calibre.rules' as rules file
parse_file: reading '/lib/udev/rules.d/95-udev-late.rules' as rules file
udev_rules_new: rules use 49272 bytes tokens (4106 * 12 bytes), 14436 bytes
buffer
udev_rules_new: temporary index used 20040 bytes (1002 * 20 bytes)
udev_device_new_from_syspath: device 0xb7fde9a0 has devpath '/class/sound/dsp'
udev_device_new_from_syspath: device 0xb7fdeb20 has devpath '/class/sound/dsp'
udev_device_read_db: device 0xb7fdeb20 filled with db file data
udev_rules_apply_to_event: GROUP 29 /lib/udev/rules.d/40-alsa.rules:3
udev_device_new_from_syspath: device 0xb7fdee40 has devpath
'/devices/pci0000:00/0000:00:1b.0'
udev_device_new_from_syspath: device 0xb7fdf0a0 has devpath '/devices/pci0000:00'
udev_rules_apply_to_event: LINK 'char/14:3' /lib/udev/rules.d/50-udev-
default.rules:5
udev_rules_apply_to_event: RUN 'socket:@/org/freedesktop/hal/udev_event'
/lib/udev/rules.d/90-hal.rules:2
udev_event_execute_rules: no node name set, will use kernel name 'dsp'
udev_device_update_db: unable to create db file
'/dev/.udev/db/\x2fclass\x2fsound\x2fdsp': Permission denied
udev_node_add: creating device node '/dev/dsp', devnum=14:3, mode=0660, uid=0,
gid=29
udev_node_mknod: preserve file '/dev/dsp', because it has correct dev_t
update_link: '/dev/char/14:3' with target '/dev/dsp' has the highest priority 0,
create it
node_symlink: preserve already existing symlink '/dev/char/14:3' to '../dsp'
udevadm_test: run: 'socket:@/org/freedesktop/hal/udev_event'
```

كما هو واضح , المكتبات التي يتعامل معها جهاز dsp و الأجهزه التي يعتمد عليها أيضاً , بإمكانك عن طريق القراءه المُتأنيه للمخرجات الحصول على نتائج مُرضيه لقواعدك .

الخاتمه

بهذا قد تم الإنتهاء من البحث النعليمي هذا و أدعو الله أن يكون قد وفقني في تغطية أكبر كم ممكن من طرق التعامل مع الـ Udev و على أمل من الله اني أكون قد قمت بتوعية أخوتي العرب من مخاطر الوقوع في لبس أسماء الأجهزه و تقديم حلول لعدم الوقوع في هذا الخطأ الذي قد يُأدي إلى خسارة الكثير من المعلومات الهامه .

وفي النهايه أتمنى من كل من قراء هذا البحث أن يقوم بنشره أكثر فأكثر حتى تعم الفائده علينا جميعاً و إذا أمكن الإضافه على معلوماته فأكون شاكر له بعد الله

م/ سيف أباظة

المراجع

- Wiki : /en.wikipedia.org
- Daniel Drake : reactivated.net
- www.togaware.com